Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars

Luke Zettlemoyer and Michael Collins
MIT CSAIL

The Problem

Learning to Map Sentences to Logical Form

`Texas borders Kansas`

borders(*texas*,*kansas*)

CSAIL

Several potential applications

- Natural Language Interfaces to Databases
- Dialogue Systems
- Machine Translation

CSAIL

Some Training Examples

Input: What states border Texas?

Output: $\lambda x.state(x) \wedge borders(x,texas)$

Input: What is the largest state?

Output: $argmax(\lambda x.state(x), \lambda x.size(x))$

Input: What states border the largest state?

Output: $\lambda x.state(x) \wedge borders(x, argmax(\lambda y.state(y), \lambda y.size(y)))$

CSAIL

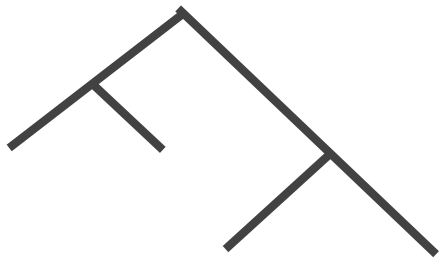
Our Approach

Learn lexical information (syntax/semantics) for words:

- `Texas` | syntax = noun phrase (`NP`) : semantics = *`texas`*
- `states` | syntax = noun (`N`) : semantics = *`λx.state(x)`*

Learn to parse to logical form:

Input: `What states border Texas?`



Output: *`λx.state(x) ∧ borders(x,texas)`*

CSAIL

Background

- Combinatory Categorial Grammar (CCG)
  - Lexicon
  - Parsing Rules (Combinators)
- Probabilistic CCG (PCCG)

CSAIL

CCG Lexicon

Words	Category
	Syntax : Semantics
Texas	NP : *texas*
borders	(S\NP)/NP : $\lambda x.\lambda y.borders(y,x)$
Kansas	NP : *kansas*
Kansas city	NP : *kansas_city_MO*

CSAIL

Parsing Rules (Combinators)

- Application
  - `X/Y : f` `Y : a` => `X : f(a)`

(S\NP)/NP	**NP**	**S\NP**
λx.λy.borders(y,x)	***texas***	***λy.borders(y,texas)***

  - `Y : a` `X\Y : f` => `X : f(a)`

NP	**S\NP**	**S**
kansas	***λy.borders(y,texas)***	***borders(kansas,texas)***

- Additional rules
  - Composition
  - Type Raising

CSAIL

CCG Parsing

Texas	borders	Kansas
NP *texas*	(S\NP)/NP *λx.λy.borders(y,x)*	NP *kansas*
	S\NP *λy.borders(y,kansas)*	
	S *borders(texas,kansas)*	

CSAIL

Parsing a Question

What	states	border	Texas
S/(S\NP)/N	N	(S\NP)/NP	NP
$\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$	$\lambda x.state(x)$	$\lambda x.\lambda y.borders(y,x)$	$texas$
S/(S\NP)		S\NP	
$\lambda g.\lambda x.state(x) \wedge g(x)$		$\lambda y.borders(y,texas)$	
S			
$\lambda x.state(x) \wedge borders(x,texas)$			

CSAIL

Probabilistic CCG (PCCG)

Log-linear model:

- A CCG for parsing
- Features
  - $f_i(L, S, T)$: number of times lexical item i is used in the parse T that maps from sentence S to logical form L
- A parameter vector θ with an entry for each f_i

CSAIL

PCCG Distributions

Log-linear model:

- Defines a joint distribution:

$$P(L,T \mid S;\theta) = \frac{e^{f(L,T,S)\cdot\theta}}{\sum_{(L,T)} e^{f(L,T,S)\cdot\theta}}$$

- Parses are a hidden variable:

$$P(L \mid S;\theta) = \sum_{T} P(L,T \mid S;\theta)$$

CSAIL

Learning

- Generating Lexical Items
- Learning a complete PCCG

CSAIL

Lexical Generation

Input Training Example

Sentence: `Texas borders Kansas`

Logic Form: *borders*(*texas*,*kansas*)

Output Lexicon

Words	Category
Texas	NP : *texas*
borders	(S\NP)/NP : $\lambda x.\lambda y.$*borders*(*y*,*x*)
Kansas	NP : *kansas*
...	...

CSAIL

GENLEX

- Input: a training example (S_i, L_i)
- Computation:
  1. Create all substrings of words in S_i
  2. Create categories from L_i
  3. Create lexical entries that are the cross product of these two sets
- Output: Lexicon Λ

CSAIL

Step 1: GENLEX Words

Input Sentence:

```
Texas borders Kansas
```

Ouput Substrings:

```
Texas
borders
Kansas
Texas borders
borders Kansas
Texas borders Kansas
```

CSAIL

Step 2: GENLEX Categories

Input Logical Form:

`borders(texas,kansas)`

Output Categories:

...

...

...

CSAIL

Two GENLEX Rules

Input Trigger	Output Category
a constant c	NP : c
an arity two predicate p	(S\NP)/NP : $\lambda x.\lambda y.p(y,x)$

Example Input: *borders(texas,kansas)*

Output Categories:

NP : *texas*, NP : *kansas*,

(S\NP)/NP : $\lambda x.\lambda y.$*borders*(y,x)

CSAIL

All of the Category Rules

Input Trigger	Output Category
a constant c	NP : c
arity one predicate p	N : $\lambda x.p(x)$
arity one predicate p	S\NP : $\lambda x.p(x)$
arity two predicate p	(S\NP)/NP : $\lambda x.\lambda y.p(y,x)$
arity two predicate p	(S\NP)/NP : $\lambda x.\lambda y.p(x,y)$
arity one predicate p	N/N : $\lambda g.\lambda x.p(x) \wedge g(x)$
arity two predicate p and constant c	N/N : $\lambda g.\lambda x.p(x,c) \wedge g(x)$
arity two predicate p	(N\N)/NP : $\lambda x.\lambda g.\lambda y.p(y,x) \wedge g(x)$
arity one function f	NP/N : $\lambda g.argmax/min(g(x),\lambda x.f(x))$
arity one function f	S/NP : $\lambda x.f(x)$

CSAIL

Step 3: GENLEX Cross Product

Input Training Example

Sentence: `Texas borders Kansas`

Logic Form: *borders*(*texas*,*kansas*)

Output Lexicon

Output Substrings:

- `Texas`
- `borders`
- `Kansas`
- `Texas borders`
- `borders Kansas`
- `Texas borders Kansas`

X

Output Categories:

`NP : texas`
`NP : kansas`
`(S\NP)/NP :`
$\lambda x.\lambda y.borders(y,x)$

GENLEX is the cross product in these two output sets

GENLEX: Output Lexicon

Words	Category
Texas	NP : *texas*
Texas	NP : *kansas*
Texas	(S\NP)/NP : $\lambda x.\lambda y.borders(y,x)$
borders	NP : *texas*
borders	NP : *kansas*
borders	(S\NP)/NP : $\lambda x.\lambda y.borders(y,x)$
...	...
Texas borders Kansas	NP : *texas*
Texas borders Kansas	NP : *kansas*
Texas borders Kansas	(S\NP)/NP : $\lambda x.\lambda y.borders(y,x)$

CSAIL

A Simple Algorithm

Inputs:

Initial lexicon Λ_0

The initial lexicon has two types of entries:

- Domain Independent:

  Example:

  `What | S/(S\NP)/N :` $\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$

- Domain Dependent:

  Example:

  `Texas | NP :` *texas*

A Simple Algorithm

Inputs:

Initial lexicon Λ_0

Training examples $E = \{(S_i, L_i) : i = 1 \ldots n\}$

Initialization:

Create lexicon $\Lambda^* = \Lambda_0 \cup \bigcup_{i=1}^{n} GENLEX(S_i, L_i)$

Create features f

Create initial parameters θ^0

Computation:

Estimate parameters $\theta = STOCGRAD(E, \theta^0, \Lambda^*)$

Output:

PCCG (Λ^*, θ, f)

CSAIL

The Final Algorithm

Inputs: Λ_0, E

Initialization: Create Λ^*, f, θ^0

Computation:

For $t = 1 \ldots T$

1. Prune Lexicon:
   - For each $(S_i, L_i) \in E$
     - Set $\lambda = \Lambda_0 \cup GENLEX(S_i, L_i)$
     - Calculate $\pi = MAXPARSE(S_i, L_i, \lambda, \theta^{t-1})$, the set of highest scoring correct parses
     - Define λ_i to be lexical items in a parse in π
   - Set $\Lambda^t = \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$
2. Estimate parameters: $\theta^t = STOCGRAD(E, \theta^{t-1}, \Lambda^t)$

Output: PCCG (Λ^T, θ^T, f)

CSAIL

Related Work

- CHILL (Zelle and Mooney, 1996)
  - learns deterministic parser; assumes semantic lexicon as input (`borders | borders(_,_)`)
- WOLFIE (Thompson and Mooney, 2002)
  - learns complete lexicon; deterministic parsing
- COCKTAIL (Tang and Mooney, 2001)
  - best results; statistical parsing; assumes semantic lexicon

CSAIL

Experiments

Two database domains:

- Geo880
  - 600 training examples
  - 280 test examples
- Jobs640
  - 500 training examples
  - 140 test examples

CSAIL

Evaluation

Test for completely correct semantics

- Precision:

  # correct / total # parsed

- Recall:

  # correct / total # sentences

CSAIL

Results

	Geo 880		Jobs 640	
	Precision	Recall	Precision	Recall
Our Method	96.25	79.29	97.36	79.29
COCKTAIL	89.92	79.40	93.25	79.84

Example Learned Lexical Entries

Words	Category
states	N : $\lambda x.state(x)$
major	N/N : $\lambda g.\lambda x.major(x) \wedge g(x)$
population	N : $\lambda x.population(x)$
cities	N : $\lambda x.city(x)$
river	N : $\lambda x.river(x)$
run through	(S\NP)/NP : $\lambda x.\lambda y.traverse(y,x)$
the largest	NP/N : $\lambda g.argmax(g,\lambda x.size(x))$
rivers	N : $\lambda x.river(x)$
the highest	NP/N : $\lambda g.argmax(g,\lambda x.elev(x))$
the longest	NP/N : $\lambda g.argmax(g,\lambda x.len(x))$
...	...

CSAIL

Error Analysis

Low recall: GENLEX is not general enough

- Fails to parse 10% of training examples

Some unparsed examples include:

- Through which states does the Mississippi run?
- If I moved to California and learned SQL on Oracle could I find anything for 30000 on Unix?

CSAIL

Future Work

- Improve recall
- Explore robust parsing techniques for ungrammatical input
- Develop new domains
- Integrate with a dialogue system

The End

Thanks

Convergence

Some Guarantees

1. Prune Lexicon
   - Will not decrease accuracy on training set
2. Estimate parameters
   - Should increase the likelihood of the training set

CSAIL